

An Iterative Local Search Based hybrid algorithm for the service area problem

Yunfeng Kong^{a,b}

^a Key Laboratory of Geospatial Technology for the Middle and Lower Yellow River Regions, Ministry of Education, Henan University, Kaifang, China;

^b College of Environment and Planning, Henan University, Kaifang, China.

Abstract: This article presents a hybrid algorithm for the service area problem. The design of service areas is one of the essential issues in providing efficient services in both the public and private sectors. For a geographical region with a number of small spatial units, the service area problem is to assign the service-demand units to the service-supply units such that each facility has a service area. The basic criteria for the service areas are the highest service accessibility, the contiguous service areas, and that the service demand does not exceed the service supply in each service area. A hybrid algorithm for the service area problem is proposed by extending iterative local search (ILS) algorithm with three schemes: population-based ILS, variable neighborhood descent (VND) search, and set partitioning. The performance of the algorithm was tested using 60 well-designed instances. Experimentation showed that the instances could be solved effectively and efficiently. The solutions found by the hybrid algorithm approximate optimal solutions or the lower bounds with an average gap of 0.15%.

Keywords: service area problem; hybrid algorithm; local search; set partitioning

1. Introduction

This article deals with the service area problem (SAP). The design of service areas is one of the essential issues in providing efficient services in both the public and private sectors (Daskin 2011). The delineation of service areas for schools (Ferland and Guenette 1990; Schoepfle and Church 1991; Caro *et al.* 2014), hospitals and healthcare centers (Pezzella *et al.* 1981; Jacobs *et al.* 1996; Emiliano *et al.* 2017; Hu *et al.* 2018), disaster shelters (Li *et al.* 2008; Hu *et al.* 2014), green energy resources (Yanik *et al.* 2016) and many other facilities can be generalized as a contiguity-constrained capacitated facility SAP. For a geographic area with small spatial units, the SAP should assign the service-demand units to service-supply units such that each facility has a service area and some criteria are satisfied. The basic criteria for the design of service areas are the highest service accessibility, the

contiguous service areas, and that the service demand does not exceed the service supply in each service area. Service accessibility can be evaluated by total travel distance from demand units to their supply units. The shortest travel distance is usually preferred when using the service. Contiguous service area is also a necessity for satisfying policy or management purposes. In addition, the total service demand in each service area should be no greater than its service capacity.

The SAP can be defined as a contiguity-constrained generalized assignment problem (GAP). It aims to minimize the total travel distance when using the service while satisfying constraints on facility capacity and service area contiguity. The GAP is known to be nondeterministic polynomial time hard (NP-hard). The constraints on spatial contiguity also pose great obstacles in modeling and solving the geographic problems (Duque *et al.* 2011). Consequently, various exact and heuristic methods have been proposed for solving the SAP.

Mixed integer linear programming (MILP) has been widely utilized to solve the districting problems. Owing to the computational complexity of the districting problems, many of the MILP models developed since the 1960s were simplified by ignoring some districting constraints such as district contiguity and unit integrity. The model solutions must be adjusted by repairing split units and fragmented districts. Such methods have been suggested for school districting (Koenigsberg 1968; Franklin and Koenigsberg 1973; Caro *et al.* 2004), political districting (Hess *et al.* 1965; Hojati 1996; George *et al.* 1997; Li *et al.* 2007), and the territory design problem (Kalcsics *et al.* 2005). Modeling and repairing approaches could be adapted to solving the SAP. However, in case of a spatial mismatch between service demands and supplies in a geographic region, it may be difficult to find satisfactory solutions.

The second class of MILP models have considered district contiguity. Three types of formulations on district contiguity—tree-based, order-based, and flow-based—can be embedded into an assignment model or a location-allocation model for solving districting problems (Shirabe 2005; Duque *et al.* 2011). Nemoto and Hotta (2003) proposed a MILP model for political districting. The model considers district contiguity, but it does not guarantee the compactness of the districts. Salazar-Aguilar *et al.* (2011) proposed a bi-objective model for designing commercial territories. Constraints such as balance of sales volume and district contiguity are formulated in the model. However, four hours of computation time were used to solve the problem instances with 150 units and 6 territories. Plane *et al.* (2019) proposed a minimum inter-person separation model for political districting. For instances with 6×6 grids can be solved optimally or near-optimally in 6.30-53,632.22 seconds. Kong *et al.* (2019) formulated a center-based allocation model with flow-based constraints on district contiguity to solve the districting problem. Given

district centers from a K-medoids algorithm, the districting instances could be solved efficiently. Since the service-supply units are already provided in the SAP, the model could be modified to solve the problem.

The third exact approach has been suggested based on the set partitioning problem (SPP). Given a large number of feasible districts generated by the construction or heuristic methods, the SPP model attempts to select an optimal set of districts from the candidate districts as a districting solution (Garfinkel and Nemhauser 1970; Nygreen 1988; Mehrotra *et al.* 1998). Solutions from the problem instances indicate that the SPP model for the small-sized instances can be solved exactly. The model results also seem to be satisfactory for large problems. To obtain satisfying solutions, they suggested different techniques to generate promising candidate districts. Kong *et al.* (2017) proposed an iterated local search algorithm with set partitioning to solve the school districting problem. The SPP model could choose a much better solution from the historical districts identified by the local search. However, the algorithm was not fully tested on general SAP instances.

Local-search-based and population-based heuristics are the mainstream methods used for solving the SAP. Starting from an initial solution, the local-search based algorithms, such as greedy search, simulated annealing, tabu search, old bachelor acceptance search, and the greedy randomized adaptive search procedure (GRASP), could iteratively improve the incumbent solution using neighborhood search (Li *et al.* 2008; Ricca and Simeone 2008; Rios-Mercado and Fernandez 2009). The neighborhood space of the incumbent solution is explored by one or several search operators, and the incumbent solution is updated according to the acceptance criteria adopted in a specific algorithm. The one-unit shift is a widely used operator that moves a boundary unit to its neighboring district while maintaining the connectivity of its original district (Tavares-Pereira *et al.* 2007; Ricca and Simeone 2008; Xiao 2008; Li *et al.* 2008; Liberatore and Camacho-Collados 2016). Butsch *et al.* (2014) introduced three operators for districting problems: shift, double shift, and swap. Kong *et al.* (2017) suggested much more complicated local search operators for school districting. However, algorithms with expensive local search operators have not fully tested in the existing literature.

Population-based heuristics have become popular for solving districting problems in recent literature. The evolutionary algorithms (Forman and Yue 2003; Bergey *et al.* 2003; Bacao *et al.* 2005; Tavares-Pereira *et al.* 2007; Chou *et al.* 2007; Tavares-Pereira *et al.* 2007; Xiao 2008; Chou 2011; Hu *et al.* 2014; Liu *et al.* 2016) maintain and improve multiple candidate solutions using mechanisms such as crossover, mutation, and selection. The crossover operation for districting problems is usually implemented by overlaying two individuals and then repairing the fragmented overlaid areas (Chou *et al.* 2007; Datta *et al.* 2008; Xiao 2008; Liu *et al.* 2016). The mutation operation usually moves every boundary

unit to its neighboring districts with a small mutation possibility. To increase the speed of the algorithm convergence and the solution quality, Tavarespereira *et al.* (2007) proposed an evolutionary algorithm with local search for the districting problem. The population diversity in evolutionary algorithms and the search intensity in local search heuristics are balanced in the hybrid algorithm. In addition, nature-inspired metaheuristics such as scatter search (Salazar-Aguilar *et al.* 2012) and artificial bee colony (Rincón-García *et al.* 2015) have also been applied to solve districting problems.

These are some limitations to the solution methods for the districting problems. First, the existing algorithms are capable of solving artificial or real problem instances; however, the performance of the algorithms has not been sufficiently tested on benchmark instances, so the optimality of solutions from the algorithms is still unknown. Second, some innovative ideas in operations research, such as hybrid metaheuristics, matheuristics, learning-based adaptive algorithms, and hyper-heuristics, have not been adequately utilized for solving the districting problems. Third, some studies report that districting problems are difficult to directly solve using exact methods. However, along with progress in MILP, especially the embedded heuristics in mixed-integer programming (MIP) optimizers, state-of-the-art MIP solvers could solve increasingly difficult problems (Lodi 2017). Designing new algorithms based on the framework of modern MIP solvers may be a promising approach to the districting problems.

This article presents a hybrid algorithm for the SAP. It was proposed by extending iterative local search (ILS) algorithm with three schemes: population-based ILS, variable neighborhood descent (VND) search, and set partitioning. The algorithm was tested using 60 well-designed problem instances. The effectiveness and efficiency of the proposed hybrid algorithm for the SAP were verified using the problem instances. Compared with exact solutions, the solutions from the hybrid algorithm approximate optimal or the lower bounds with an average gap of 0.15%. To the best of authors' knowledge, it is the first time to introduce a population-based ILS with VND search and set partitioning for the SAP.

2. Problem formulation

For a geographic area, let $V = \{1, 2 \dots n\}$ be a set of n small units. Each unit i has service demand p_i . Let $S = \{s_1, s_2 \dots s_K\}$ ($S \subset V$) be a set of K service-supply units, and each unit s_k has service capacity q_k . Let d_{ik} be the distance between demand unit i and supply unit k . Let a_{ij} indicate whether units i and j share a border and N_i be a set of units that are adjacent to unit i ($N_i = \{j | a_{ij} = 1\}$).

A MILP model can be formulated by defining the SAP as a contiguity-constrained GAP. The contiguity of service areas can be ensured by a network flow model (Shirabe 2005; Duque *et al.* 2011). Let $x_{ik} \in \{0,1\}$ denote whether unit i is assigned to supply unit

s_k , H_k ($H_k \geq 0$) be the service overload of supply unit s_k , and f_{ijk} ($f_{ijk} \geq 0$) be the flow from unit i to unit j in service area k . The model for the SAP is formulated as follows:

$$\text{Minimize} \quad \alpha \sum_{k \in S} H_k + \sum_{i \in V} \sum_{k \in S} p_i d_{ik} x_{ik} \quad (1)$$

$$\text{Subject to:} \quad \sum_{k \in S} x_{ik} = 1, \forall i \in V \quad (2)$$

$$\sum_{i \in V} p_i x_{ik} \leq q_k + H_k, \forall k \in S \quad (3)$$

$$f_{ijk} \leq (n - K) x_{ik}, \forall i \in V, j \in N_i, \forall k \in S \quad (4)$$

$$f_{ijk} \leq (n - K) x_{jk}, \forall i \in V, j \in N_i, \forall k \in S \quad (5)$$

$$\sum_{j \in N_i} f_{ijk} - \sum_{j \in N_i} f_{jik} \geq x_{ik}, \forall i \in V \setminus S, \forall k \in S \quad (6)$$

$$x_{ik} = \{0, 1\}, \forall i \in V, k \in S \quad (7)$$

$$f_{ijk}, H_k \geq 0, \forall i \in V, j \in N_i, k \in S \quad (8)$$

The objective function (1) is to minimize the total travel distance from service demand units to their supply units. It also penalizes service overloads by using a large enough coefficient α . Constraints (2) confirm that each unit i is assigned to only one service-supply unit. Constraints (3) are soft constraints on service capacities. Constraints (4), (5), and (6) are the flow-based formulations on contiguity, which are rewritten formulations in Duque *et al.* (2011). For two adjacent units i and j , if both of them are assigned to service-supply unit s_k ($x_{ik}=1$ and $x_{jk}=1$), there might be a flow from unit i to j in basin k ($f_{ijk} \geq 0$); otherwise, there is no any flow between them ($f_{ijk} = 0$). This is confirmed by constraints (4) and (5). Constraints (6) ensure that one unit of flow must be created at non-service-supply unit i if it is assigned to service-supply unit s_k . Constraints (4), (5) and (6) enforce all the flows created in service area of k must run to the sink unit k . Constraints (7) and (8) impose restrictions on decision variables. Note that the soft constraints (3) on facility capacities are used in the model. The service overloads (H_k) in the constraints are penalized by the objective function. There are two advantages of using soft constraints. First, the soft constraints could help the MIP optimizer to find a feasible solution efficiently and guide the search toward preferred solutions. Second, using hard capacity constraints, there is no feasible solution for some instances, e.g. the total demand is greater than the total supply, or both the assignment constraints and capacity constraints cannot be satisfied. Such infeasible instances can be handled by the soft constraints.

3. A hybrid algorithm

3.1 Iterative local search

A hybrid algorithm for the SAP was proposed based on ILS algorithm. ILS is a simple, easy to implement, and quite effective metaheuristic for discrete optimization problems (Lourenço *et al.* 2010). ILS starts from an initial solution and iteratively improves the

solution by local search and perturbation. Local search heuristic is used to intensively explore the solution space. However, the iterative search may easily get trapped in local optima that are far away from the global optimum. ILS escapes from local optima by applying perturbations to the current local minimum. The basic ILS procedure is illustrated in Algorithm 1.

Algorithm 1: Iterated local search

1. $s_0 = \text{GenerateInitialSolution}();$
 2. $s = \text{LocalSearch}(s_0);$
 3. Repeat until termination condition met:
 4. $s' = \text{Perturbation}(s);$
 5. $s^* = \text{LocalSearch}(s');$
 6. If $f(s^*) < f(s)$: $s = s^*;$
 7. Output s .
-

3.2 Initial solution

The transportation problem (TP) model as follows was used to generate initial solutions. The model assigns the demand in each unit to one or multiple supply units with minimum traveling cost. The TP can be easily solved by a MIP solver owing to its bipartite structure. The small random coefficients ϵ_{ik} ($|\epsilon_{ik}| < 0.02$) are used in the objective function to obtain a random solution.

$$\text{Minimize} \quad \sum_{i \in V} \sum_{k \in S} (1 + \epsilon_{ik}) p_i d_{ik} x_{ik} \quad (9)$$

$$\text{Subject to:} \quad \sum_{k \in S} x_{ik} = 1, \forall i \in V \quad (10)$$

$$\sum_{i \in V} p_i x_{ik} \leq q_k, \forall k \in S \quad (11)$$

$$x_{ik} = [0, 1], \forall i \in V, k \in S \quad (12)$$

The solution from the TP model needs to be repaired. First, some units in a solution may be split into two or more parts. For each split unit, the algorithm assigns it to a service area according to the largest portion of split. Second, some service areas in a solution may be non-contiguous. The non-contiguous areas can be repaired by deleting the fragmented units and then reassigning them to their neighboring areas. Note that the solution from region growth may violate the constraints on service capacities. The infeasible solutions are allowed in the algorithm. However, a solution with service overloads will be penalized in the following local search by the objective function (1) and tend to be a feasible solution.

3.3 Local search

The design of local search operators in optimization algorithm is critical for repeatedly improving the incumbent solution. The one-unit shift is a widely used operator that moves a boundary unit to its neighboring district while maintaining the connectivity of its original district (Tavares-Pereira *et al.* 2007; Ricca and Simeone 2008; Xiao 2008; Li *et al.* 2008;

Liberatore and Camachocollados 2016). Butsch *et al.* (2014) introduced three operators for districting problems: shift, double shift, and swap.

Two local search operators are used in the algorithm to improve the solutions. The operators attempted to move one or more units located on the boundary to their neighboring service areas. Note that only the feasible moves are allowed, because when moving a boundary unit from its original area to a destination area, the original area may be non-contiguous. The one-unit shift and two-unit shift are illustrated in Figure 1.

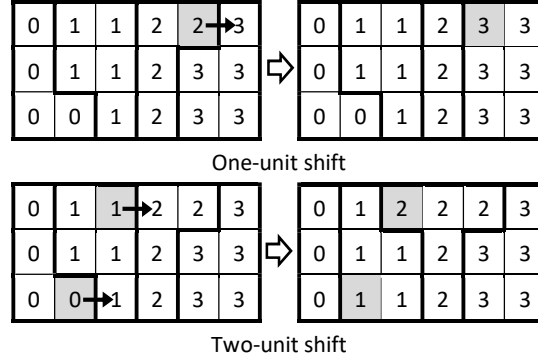


Figure 1: Examples of the local search operators

The one-unit shift moves boundary unit i to one of its neighboring area k as outlined in Algorithm 2. In the algorithm, all the boundary units are selected and then shuffled randomly. For each boundary unit i , the algorithm tries to move the boundary unit from its original area to one of its neighboring areas. The move is accepted in cases that the original area is connective, and the new solution is better than the incumbent solution.

Algorithm 2: one-unit shift

1. $ulist = \text{BoundaryUnits}(s);$
 2. $ulist = \text{Shuffle}(ulist);$
 3. For i in $ulist$:
 4. For k in $\text{NeighboringAreas}(i)$:
 5. $s^* = \text{Move}(s, i, k);$
 6. If $\text{Area}(i)$ is contiguous and $f(s^*) < f(s)$:
 7. $s = s^*;$
 8. Return s^* .
-

The two-unit shift moves boundary unit i to one of its neighboring areas k , and at the same time moves boundary unit j in area k to one of its neighboring areas (Algorithm 3). In other words, one unit is moved into area k , and another unit in area k is moved out. Differing from the one-unit shift that involves two areas, a two-unit shift usually involves three areas. Swapping two units between two adjacent areas is a special case that involves two areas.

Algorithm 3: two-unit shift

1. $ulist = \text{BoundaryUnits}(s);$
 2. $ulist = \text{Shuffle}(ulist);$
-

-
3. For i in $ulist$:
 4. For j in $ulist$:
 5. If $Area(i)$ and $Area(j)$ are not adjacent: continue;
 6. For k in $NeighboringAreas(j)$:
 7. $s^* = \text{Move}(s, i, j, k)$;
 8. If $Area(i)$ or $Area(j)$ is non-contiguous: continue;
 9. If $f(s^*) < f(s)$: $s = s^*$;
 10. Return s .
-

The two local search operators have different search space and computational complexity: $O(Kn)$ and $O(Kn^2)$. However, the number of possible moves is generally much fewer, because only the boundary units can be moved to their neighboring areas, and there are only a few neighboring areas available for a boundary unit.

3.4 Perturbation

Perturbation is one of the key components of ILS algorithm. Using local search operators, the solution may easily reach to local optima. A ruin and recreate procedure is useful to perturb the SAP solution from local optima. There are multiple methods to ruin the solution such as deleting some boundary units, deleting all the units in some areas, and deleting some units in a connective region. Then, the deleted units can be reassigned to their nearest facilities. The second perturbation method is to move some boundary units to their neighboring areas. A repair procedure on the new solution is usually necessary since some areas in the perturbed solution may be non-contiguous. On the other hand, the objective of perturbed solution may become worse. However, the following local search could improve the solution quality significantly.

The strength of perturbation is critical for ILS algorithm. A very strong perturbation tends to generate a much worse solution, such that better solutions will be found with a very low probability. On the other hand, for a very small perturbation, the solution will often back to its original state of local optimum by the following local search. An appropriate strength of perturbation will maintain the diversification of ILS search, and thus benefits to find better solutions.

Four perturbation schemes are randomly used in our ILS algorithm: ruin boundary units and recreate, ruin service areas and recreate, ruin a connective region and recreate, and move boundary units to their neighboring areas. Let *strength* be the strength of perturbation, i.e. a percentage of the solution components, for example, *strength%* of boundary units, *strength%* of service areas, or *strength%* of all units. In all perturbation schemes, the perturbed units/areas are randomly selected.

3.5 Set partitioning

Let Ω denote the set of historical service areas identified in ILS algorithm. Each area i has an objective c_i and a set of units U_i in the area. Let subset $\Omega_j = \{i | i \in \Omega, j \in U_i\}$, the SPP model (13)-(15) could be used to select a subset service areas of Ω , which gives a minimal objective and also covers all the units in set V . The decision variable x_i indicates whether the candidate area i is selected.

$$\text{Minimize } \sum_{i \in \Omega} c_i x_i \quad (13)$$

$$\text{Subject to: } \sum_{i \in \Omega_j} x_i = 1, \forall j \in V \quad (14)$$

$$x_i \in \{0,1\}, \forall i \in \Omega \quad (15)$$

The set partitioning could be added to ILS algorithm as a post procedure aiming to improve the SAP solution. The SPP is known to be NP-hard; however, existing MIP optimizers could solve the SPP instances with considerable sizes of Ω and V .

3.6 Contiguity of service areas

To keep the contiguity of service areas in SAP solutions, several algorithms are frequently used in local search and perturbation. The first algorithm is to check the connectivity of a service area. If a spanning tree can be established using all the units in an area, the area is connective (Liu *et al.* 2016). The second algorithm is to find the fragmented units in a solution. The spanning tree method can also identify the fragmented units. For each service area, a spanning tree is gradually constructed from the supply unit. If some units cannot be assigned to the tree, they are the fragmented units. The third algorithm is to repair a solution with non-contiguous service areas. The fragmented service areas are repaired by deleting the fragmented units and then gradually reassigning them to their neighboring areas.

3.7 A hybrid algorithm

We introduced a hybrid algorithm based on ILS to solve the SAP. The standard ILS algorithm was enhanced by three schemes: population-based search, variable neighborhood descent (VND) search, and set partitioning. The proposed algorithm is outlined in Algorithm 4.

Algorithm 4: Hybrid algorithm for SAP

Parameters: population size (*psize*), minimum dissimilarity between any two solutions (*mindiff*), perturbation strength (*strength*), number of consecutive loops that the best solution is not updated (*mloops*), time limit for set partitioning (*t*).

1. $P = \text{GenerateInitialSolutions}(\text{psize})$; (Section 3.2);
 2. $\text{pool} = \text{null}$;
 3. $s_{\text{best}} = \text{Best}(P)$;
-

-
4. $notImprove=0$;
 5. While $notImprove < mloops$:
 6. Select a solution s from P randomly;
 7. $s'=Perturbation(s, strength)$; (Section 3.4);
 8. $s^*=VNDsearch(s')$;
 9. If $f(s^*) < f(s_{best})$: $s_{best}=s^*$, $notImprove=0$;
 10. else: $notImprove+=1$;
 11. $pool=UpdateServiceAreaPool(pool, s^*)$;
 12. $P=UpdatePopulation(P, s^*, mindiff)$;
 13. $s=SetPartitioning(pool, t)$; (Section 3.5);
 14. Output s .
-

The hybrid algorithm maintains a number of candidate solutions. It is a population-based algorithm rather than the standard single solution-based ILS algorithm. The diversification of ILS search is guaranteed by the perturbation scheme. However, it is usually difficult to select a strength parameter for solution perturbation. Furthermore, the best perturbation strength for an instance might depend on the perturbation scheme and the attributes of the instance. The population-based extension of ILS may enhance the diversification of local search. A number of elite and diverse solutions are selected by the algorithm. In step 12, the population is updated according to the solution objectives and the similarities among the solutions. To maintain solution diversity, only elitist and dissimilar solutions are selected to be the new population. The updated population should be dissimilar. For any two solutions, their dissimilarity can be defined as the percentage of the demand units that are assigned to different facilities. Based on this definition, a dissimilar population means that the dissimilarity between any two solutions in the population is greater than a minimum dissimilarity. This percentage can be predefined as an algorithm parameter ($mindiff$). In some cases, the size of the new population may be smaller than the parameter $popsize$; however, the following process of perturbation could generate new dissimilar solutions.

The second scheme is to use VND local search instead of simple local search in ILS. A simple local search is to improve the solution by performing one-unit shift and two-unit shift respectively. VND is repeatedly exploring different neighborhood structures by deterministic change of neighborhoods until the solution cannot be improvement. Using VND search, ILS algorithm can easily find solutions in local optimum, and then moves the solution to new search point by perturbation.

The third scheme is to improve ILS solution by set partitioning. In each ILS loop, all the service areas in solution s^* are recorded in the list $pool$. Each areal record includes the units in the service area and its objective value. At the end of ILS, thousands or more areas will be recorded in the pool. A SPP model (Section 3.5) could be used to select a better solution from the candidate service areas.

4. Experiment

4.1 Experiment settings

Three study areas, ZY, GY, and GY2 in Kong et al. (2019), were adapted for delineating the service areas. There are 324, 297 and 1276 spatial units in the study areas, respectively. Figure 2 shows the spatial units and the service demand in each unit.

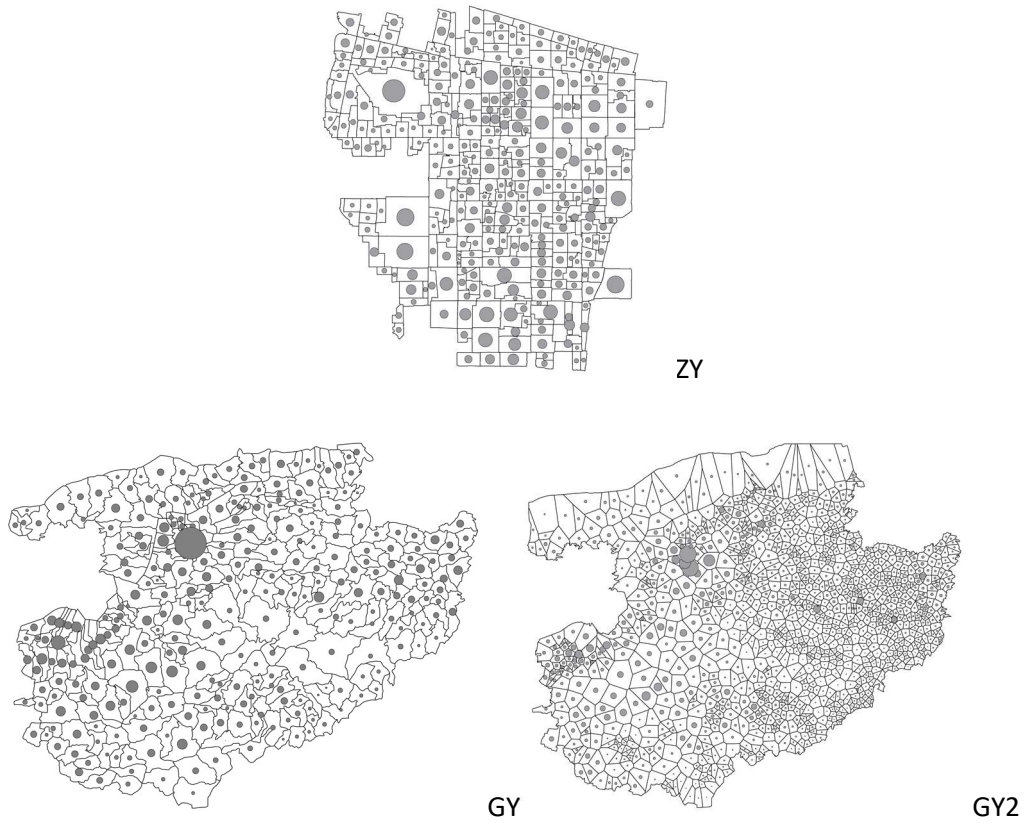


Figure 2: The spatial units in areas ZY, GY and GY2 (Kong et al. 2019). The grey circles indicate the size of the service demand in each unit.

Using the three study areas, 60 instances for the SAP were prepared to test the hybrid algorithm. For each study area, 20 instances were designed as follows. (1) 36, 44 and 24 units in the three areas were assumed to be the candidates of service-supply units with service-supplies of 9195, 42326 and 1324763, respectively. (2) Select a number of facilities from candidates randomly. 13-17, 37-41, and 18-22 service-supply units were selected for the three areas, respectively. (3) Adjust the facility capacities in each instance proportionally to ensure that the supply-demand ratio is around 1.15 and 1.03, respectively. (4) Select a number of facilities from candidates by solving a capacitated p-median problem, and then adjust the facility capacities with the same supply-demand ratio in step (3).

Table 1 shows the attributes of the instances. The 20 instances for each study area can be classified into 4 sets: random facility locations with higher supply-demand ratio (A), p-median facility locations with higher supply-demand ratio (B), random facility locations with lower supply-demand ratio (C), and p-median facility locations with lower supply-demand ratio (D). As a result, the instances are diverse in terms of the number of demand units, the number of facilities, the facility locations, the supply-demand ratio and the average number of units served by one facility.

Table 1. Supply-demand ratios of 12 sets of instances

Area	No. units	Demand	No. facilities	Set A	Set B	Set C	Set D
ZY	324	3873	13-17	1.15	1.15	1.03	1.03
GY	297	36,824	37-41	1.15	1.15	1.03	1.03
GY2	1276	819,812	18-22	1.15	1.15	1.03	1.03

The proposed algorithm was implemented by using the Python programming language. The Python script were run in PyPy 6.0, a fast and compliant implementation of the Python language (see <http://pypy.org>). Each instance was repeatedly solved for ten times. Since random mechanism was used in initial solution generation, local search and perturbation, different solutions will be obtained by repeatedly executing the algorithm. To verify the optimality of the solutions, the instances were also solved by the MILP model formulated in Section 2. The algorithm ran on a desktop computer with Intel Core i7-6700 CPU 3.40-GHz, 8-GB RAM and the Windows 10 operating system.

In the experiment, algorithm parameters are shown in Table 2. The coefficient α in objective function (1) was set to 10000. It is large enough to penalize the service overload in some service areas. To analyze the sensitivity of the parameter settings, the instances were also solved by changing one of the parameters. The TP model and SPP model were solved by the IBM ILOG CPLEX Optimizer 12.6.3. Additionally, two parameters were set for the CPLEX optimizer for solving for the SAP model: $MIPGap=10^{-10}$ and $Timelimit=7200$ (seconds). Note that we solve the SAP model in two stages: (1) solve a transportation problem (Section 3.2) and repair the fragmented solution; and (2) solve the SAP model with the repaired solution as an initial solution.

Table 2. Parameters for the hybrid algorithm

Area	ZY	GY	GY2
Population size	10	10	10
Minimum dissimilarity between any two solutions	5%	5%	5%
Perturbation strength	5%	5%	5%
Maximum number of consecutive loops that the best solution is not updated	500	500	200
Time limit for set partitioning	50s	50s	100s

4.2 Solution results

The solutions for 60 instances are shown in Table 3. The instance name in the table consists of the area name, the number of facilities and the type of facility configurations. For each instance, the lower bound, upper bound, and computation time obtained from the CPLEX optimizer are shown in columns *LB*, *UB*, and *Time*. The upper bounds labeled with asterisk are optimal objectives. For each instance, there are 10 solutions obtained from the hybrid algorithm. The best, average and worst objective gaps are shown in columns *Gap_{min}*, *Gap_{avg}* and *Gap_{max}* respectively. The gaps are calculated by the formula $(obj-LB)/LB \times 100\%$. Column *Dev* shows the standard deviation of the objective values. Column *Time* indicates the average computation time in seconds. Note that all the solutions are feasible in terms of capacity constraint and contiguity constraint.

Table 3. Solution results from 60 instances

Inst. set	Instance	CPLEX			Hybrid algorithm				
		LB	UB	Time/s	Gap _{min}	Gap _{avg}	Gap _{max}	Dev	Time/s
ZYA	zy_13a	2582.40	2582.40*	353.69	0.01%	0.16%	0.26%	0.08%	56.16
	zy_14a	2320.96	2320.96*	105.89	0.00%	0.04%	0.18%	0.06%	48.44
	zy_15a	2125.35	2125.35*	133.53	0.00%	0.12%	0.23%	0.09%	45.94
	zy_16a	1995.61	1995.61*	54.44	0.04%	0.11%	0.17%	0.04%	41.73
	zy_17a	1972.27	1972.27*	312.98	0.01%	0.07%	0.13%	0.04%	56.33
GYA	gy_37a	70172.91	70172.91*	147.39	0.00%	0.00%	0.00%	0.00%	23.40
	gy_38a	70357.88	70357.88*	219.98	0.00%	0.00%	0.00%	0.00%	25.10
	gy_39a	69942.63	69942.63*	1180.34	0.00%	0.00%	0.00%	0.00%	30.66
	gy_40a	70002.58	70002.58*	378.72	0.00%	0.00%	0.00%	0.00%	37.10
	gy_41a	69486.77	69486.77*	166.11	0.00%	0.04%	0.21%	0.07%	33.58
GY2A	gy2_18a	2151429.06	2153367.10	7201.48	0.18%	0.20%	0.23%	0.02%	82.50
	gy2_19a	2032040.61	2032040.61*	263.55	0.03%	0.05%	0.06%	0.01%	55.01
	gy2_20a	1974506.43	1974506.43*	295.04	0.01%	0.02%	0.05%	0.01%	56.54
	gy2_21a	1931624.28	1931624.28*	84.78	0.00%	0.02%	0.03%	0.01%	39.18
	gy2_22a	1875355.92	1875355.92*	47.84	0.00%	0.01%	0.01%	0.00%	31.94
ZYP	zy_13b	1811.21	1811.21*	107.56	0.02%	0.08%	0.15%	0.05%	38.04
	zy_14b	1731.02	1731.02*	152.00	0.00%	0.22%	0.54%	0.19%	43.18
	zy_15b	1656.53	1656.53*	78.72	0.00%	0.13%	0.51%	0.17%	27.47
	zy_16b	1647.33	1647.33*	147.52	0.00%	0.01%	0.04%	0.02%	42.52
	zy_17b	1618.93	1618.93*	153.38	0.05%	0.24%	0.85%	0.34%	47.96
GYB	gy_37b	69052.55	69052.55*	154.39	0.00%	0.00%	0.00%	0.00%	26.87
	gy_38b	68989.99	68989.99*	312.53	0.00%	0.00%	0.00%	0.00%	20.32
	gy_39b	69047.82	69047.82*	162.70	0.00%	0.00%	0.00%	0.00%	25.52
	gy_40b	68677.92	68677.92*	147.28	0.00%	0.00%	0.00%	0.00%	24.55
	gy_41b	68555.13	68555.13*	196.78	0.00%	0.02%	0.15%	0.05%	28.13

GY2B	gy2_18b	1953522.29	1953522.29*	222.22	0.00%	0.01%	0.02%	0.01%	33.28
	gy2_19b	1927032.00	1927032.00*	67.03	0.00%	0.01%	0.02%	0.00%	32.49
	gy2_20b	1904637.49	1904637.49*	74.10	0.00%	0.00%	0.01%	0.00%	37.30
	gy2_21b	1889743.21	1889743.21*	86.73	0.00%	0.01%	0.02%	0.01%	44.94
	gy2_22b	1875200.30	1875200.30*	59.46	0.00%	0.00%	0.01%	0.00%	32.68
ZYC	zy_13c	2822.23	2822.23*	43.70	0.00%	0.19%	0.40%	0.13%	60.05
	zy_14c	2589.93	2589.93*	514.55	0.07%	0.25%	0.39%	0.13%	66.04
	zy_15c	2405.40	2405.40*	399.81	0.14%	0.36%	0.69%	0.20%	76.24
	zy_16c	2260.01	2260.01*	274.39	0.11%	0.45%	0.77%	0.20%	51.74
	zy_17c	2250.94	2250.94*	575.95	0.13%	0.23%	0.35%	0.08%	57.98
GYC	gy_37c	79828.37	79924.28	7204.52	0.19%	0.29%	0.57%	0.14%	51.77
	gy_38c	79514.67	79514.67*	1112.73	0.00%	0.12%	0.69%	0.22%	55.39
	gy_39c	78748.95	78748.95*	1280.30	0.00%	0.16%	1.00%	0.32%	55.31
	gy_40c	78402.60	78402.60*	1088.69	0.00%	0.07%	0.13%	0.04%	53.01
	gy_41c	79603.48	79771.00	7203.73	0.21%	0.34%	0.74%	0.16%	68.47
GY2C	gy2_18c	2316782.75	2317477.99	7204.80	0.06%	0.13%	0.18%	0.04%	105.16
	gy2_19c	2141600.45	2142243.12	7205.44	0.14%	0.18%	0.24%	0.03%	105.21
	gy2_20c	2118415.85	2119263.56	7205.84	0.08%	0.14%	0.19%	0.04%	113.90
	gy2_21c	2026852.86	2026852.86	7205.17	0.09%	0.11%	0.14%	0.02%	89.63
	gy2_22c	2003751.38	2003951.77	7204.17	0.03%	0.10%	0.16%	0.05%	98.95
ZYD	zy_13d	2050.88	2050.88*	77.56	0.02%	0.04%	0.08%	0.02%	55.50
	zy_14d	1927.72	1927.72*	202.80	0.11%	0.33%	0.84%	0.27%	52.55
	zy_15d	1874.58	1874.58*	104.13	0.10%	0.27%	0.52%	0.14%	60.89
	zy_16d	1840.85	1840.85*	787.34	0.26%	0.65%	1.00%	0.26%	45.89
	zy_17d	1841.48	1841.48*	3965.80	0.73%	0.82%	0.99%	0.08%	56.36
GYD	gy_37d	76744.96	76744.96*	1090.05	0.00%	0.00%	0.00%	0.00%	52.91
	gy_38d	77792.10	77792.10*	2035.78	0.00%	0.02%	0.07%	0.03%	49.18
	gy_39d	78960.76	79150.72	7206.91	0.24%	0.39%	0.67%	0.16%	55.73
	gy_40d	78321.25	78572.69	7205.88	0.32%	0.47%	0.60%	0.09%	57.29
	gy_41d	80428.82	80533.51	7205.42	0.15%	0.30%	0.49%	0.10%	52.80
GY2D	gy2_18d	2016743.88	2016743.88*	6941.55	0.10%	0.13%	0.18%	0.04%	67.05
	gy2_19d	1986365.28	1986365.28*	929.58	0.06%	0.09%	0.13%	0.02%	55.23
	gy2_20d	2026789.24	2026991.94	7205.20	0.06%	0.19%	0.37%	0.08%	105.73
	gy2_21d	2016126.53	2016529.84	7208.63	0.15%	0.17%	0.21%	0.03%	104.84
	gy2_22d	2028023.26	2028631.85	7207.47	0.11%	0.13%	0.14%	0.01%	71.51

Table 3 shows that all the SAP instances can be solved optimally or near-optimally by CPLEX optimizer. Among the model solutions, 46 are optimal and 14 are near-optimal with *MIPGap* between 0.00% and 0.32%. Different from the early reports that only very small districting instances could be exactly solved, some larger instances for the SAP could be solved in our experiment. This should be attributed to the progress in MILP solver and CPU speed, but also the soft-constrained model formulations. It is also found that the instance complexity depends on both the instance size and the supply-demand ratio. The instances

with high supply-demand ratio (instance sets A and B) would be solved more efficiently than those with low supply-demand ratio (instance sets C and D).

The results in Table 3 also show that the proposed hybrid algorithm is effective and efficient. First, the solutions from the hybrid algorithm approximate to the optimal solutions or the lower bounds with an average gap of 0.15% and ranging from 0.00% to 0.82%. Among the best solutions in column Gap_{min} , 28 (46.7%) are optimal. Table 4 summarizes the number of optimal solution found by CPLEX and the hybrid algorithm. Second, the objective deviations are small, ranging from 0.00% to 0.38%. Third, all the instances were solved by the hybrid algorithm in a reasonable computation time: averages of 51.55, 41.35 and 68.15 seconds for areas ZY, GY and GY2, respectively.

Table 4. Number of optimal solutions found by CPLEX and the hybrid algorithm

Instance set	CPLEX				Hybrid algorithm			
	A	B	C	D	A	B	C	D
ZY	5	5	5	5	2	3	1	0
GY	5	5	3	2	5	5	3	2
GY2	4	5	0	2	2	5	0	0

Set partitioning in the hybrid algorithm is capable of improving the ILS solutions. The average improvement on all instances is 0.12%. Table 5 illustrates the improvements for each set of instances. For area GY, the instances of type C and D are significantly improved by set partitioning: 0.60% and 0.43%, respectively. Occasionally, the ILS heuristic may be difficult to find a feasible solution; however, the set partitioning procedure could select a feasible solutions with high quality. The computation times for solving SP models are shown in Table 6. The SP models for SAP instances could be solved very efficiently. Generally, the SAP solutions could be improved by set partitioning in a short computation time.

Table 5. Solution improvement by set partitioning

Instance set	A	B	C	D
ZY	0.04%	0.07%	0.07%	0.08%
GY	0.06%	0.04%	0.60%	0.43%
GY2	0.04%	0.00%	0.01%	0.04%

Table 6. Computation time in seconds for set partitioning

Instance set	A	B	C	D
ZY	1.53	1.06	3.33	2.60
GY	0.29	0.23	0.93	1.12
GY2	1.00	0.53	3.39	4.96

4.3 Analysis of algorithm parameters

To evaluate the sensitivity of the parameter settings, all the instances were solved by the proposed algorithm using 9 additional sets of parameters. The solution gaps are summarized in Table 7. Column *Default* shows the solution gaps from the algorithm with parameters listed in Table 2. Other columns shows the solution gaps from the algorithm by setting different parameters:

P1: default parameters, but set population size to 1 ($p_{size}=1$);

P5: default parameters, but set population size to 5 ($p_{size}=5$);

P20: default parameters, but set population size to 20 ($p_{size}=20$);

R10: default parameters, but set perturbation strength to 10% ($strength=10\%$);

R15: default parameters, but set perturbation strength to 15% ($strength=15\%$);

OP1: default parameters, but use one-unit shift search operator only;

OP2: default parameters, but use two-unit shift search operator only;

LS: default parameters, but use simple local search in ILS loops;

RG: default parameters, but use region growth method to generate initial solutions.

Table 7. Solution gaps (%) from the Hybrid algorithm using different sets of parameters

Inst. set	Default	P1	P5	P20	R10	R15	OP1	OP2	LS	RG
ZYA	0.10	0.25	0.15	0.21	0.16	0.23	0.77	0.22	0.29	0.16
GYA	0.01	0.02	0.00	0.00	0.00	0.02	0.14	0.00	0.00	0.00
GY2A	0.06	0.09	0.07	0.05	0.05	0.06	0.16	0.09	0.05	0.22
ZYB	0.14	0.20	0.22	0.14	0.08	0.04	0.97	0.04	0.46	0.13
GYB	0.00	0.01	0.00	0.00	0.00	0.00	0.02	0.00	0.00	0.00
GY2B	0.01	0.10	0.01	0.00	0.01	0.01	0.03	0.01	0.01	0.11
ZYC	0.30	0.42	0.40	0.32	0.26	0.38	1.33	0.47	0.64	0.51
GYC	0.20	0.38	0.09	0.07	0.28	0.11	2.37	0.10	0.39	0.08
GY2C	0.13	0.34	0.30	0.14	0.17	0.16	0.53	0.33	0.18	0.77
ZYD	0.42	0.95	0.41	0.67	0.33	0.45	2.63	0.40	0.99	0.42
GYD	0.24	0.26	0.04	0.10	0.07	0.08	0.82	0.14	0.37	0.09
GY2D	0.14	0.22	0.17	0.14	0.17	0.16	0.64	0.34	0.19	0.66
Average	0.15	0.27	0.15	0.15	0.13	0.14	0.86	0.18	0.30	0.25

There are several findings from the solution results in Table 7. First, solution gaps in columns Default, OP1 and OP2 show that the two-unit shift search operator is more effective than the one-unit shift; and the combination of one-unit shift and two-unit shift operators is slightly better than the two-unit shift operator. The one-unit shift is a fast local search operator, while the two-unit shift is slower but more effective. For designing the local search procedure in ILS, an issue is that whether it is best to have a fast operator or an effective one. The experimentation suggest that the best choice is to use both of the one-unit shift and two-unit shift in ILS search.

Second, solution gaps in columns Default, P1, P5 and P20 indicate that the population-based search ($Psize = 5, 10$ or 20) is better than the single-solution-based search ($Psize = 1$). Maintaining a population of elite and diverse solutions, local search in ILS could explore much larger neighborhood space, and thus have more possibility to find better solutions. It is observed that the population-based ILS performs better than the single-solution-based ILS, and the population size is not sensitive to solution quality.

Third, other parameters have some effects on the performance of the proposed algorithm, but are not very sensitive to the solution quality. The perturbation strength is a key parameter to balance the intensification and diversification of ILS search. In our algorithm, the perturbation strength between 5%~15% is appropriate for solving most instances. For ILS local search, both the simple local search and VND search could guide the algorithm to find good solutions. Since the VND search is capable to reach local optimum in solution space, the ILS with VND search might be better than the ILS with simple local search. The final solutions for the SAP slightly depend on the initial solution method. For most instances, the procedure of solving a TP model and repairing the model solution is the best method to generate initial solutions.

5 Conclusion

A new hybrid algorithm was proposed to solve the contiguity-constrained capacitated facility SAP. To the best of the authors' knowledge, it is the first time to introduce a population-based ILS with VND search and set partitioning for delineating service areas. The performance of the proposed method was intensively tested on 60 well-designed instances. Experimentation shows that the algorithm is capable of finding high-quality near-optimal solutions in a reasonable computation time.

The proposed hybrid algorithm is different from existing algorithms in some aspects. The perturbation in ILS algorithm plays an important role in escaping from local optima and an appropriate perturbation strength could guide the ILS search approaching the global optimum. The algorithm was designed based on the standard ILS algorithm, and further enhanced by three schemes: VND search, population-based ILS, and set partitioning. VND search in ILS could extensively explore the neighborhood space of the incumbent solution, and thus increases the convergence speed of ILS. The population-based ILS maintains a set of solutions. Starting from the elite and diverse solutions, the perturbation and local search in ILS have a higher possibility of finding better solutions. Additionally, the service areas explored by the local search are reused for selecting a better solution by set partitioning. In traditional set-partitioning-based heuristics, even if different techniques are suggested to generate a large number of service areas, it is still difficult to identify the promising candidates for large problem instances. In local-search-

based or population-based heuristics, a large number of solutions are discovered; however, most of them are abandoned. Historical solutions in metaheuristics should be the high-quality candidates for the set-partitioning-based approaches. In our algorithm, all the service areas identified in each iteration were recorded and finally reselected by a SPP model.

Twelve sets of benchmark instances with different sizes and complexity for SAP were introduced based on three geographic areas. The instances are diverse in terms of the number of demand units, the number of facilities, the facility locations, the supply-demand ratio and the average number of units served by one facility. The optimal or near optimal solutions for the instances were obtained from CPLEX optimizer. The instances and all the solution results shown in this article can be downloaded from <https://github.com/yfkong>. The authors believe that the benchmark instances are valuable to evaluate existing and newly proposed algorithms for the SAP.

Finally, some general issues regarding the hybrid algorithm should be investigated in further studies. The parameter settings and the adaptive adjustment of the parameters for the algorithm remain as open issues. Both the theoretical and practical analysis on this topic may inform the algorithm to solve a specific instance more efficiently. Considering that districting problems have some common criteria, it could be possible to extend the algorithm to solve other problems such as the political districting problem and location-districting problems.

References

- Bacao, F., Lobo, V., and Painho, M., 2005. Applying genetic algorithms to zone design. *Soft Computing*, 9(5), 341-348.
- Bergey, P. K., Ragsdale, C. T., and Hoskote, M., 2003. A Simulated Annealing Genetic Algorithm for the Electrical Power Districting Problem. *Annals of Operations Research*, 121(1), 33-55.
- Butsch, A., Kalcsics, J., and Laporte, G., 2014. Districting for Arc Routing. *Inform Journal on Computing*, 26(4), 809-824.
- Caro, F., et al., 2004. School redistricting: embedding GIS tools with integer programming. *Journal of the Operational Research Society*, 55(8), 836-849.
- Chou, C., Chu, Y., and Li, S., 2007. Evolutionary Strategy for Political Districting Problem Using Genetic Algorithm. *Lecture Notes in Computer Science*, 4490(4), 1163-1166.
- Chou, C., 2011. A Knowledge-based Evolution Algorithm approach to political districting problem. *Computer Physics Communications*, 182(1), 209-212.
- Daskin, M. S., 2011. *Service science*, Hoboken: John Wiley & Sons, 183-283.

- Datta, D., *et al.*, 2008. Graph partitioning through a multi-objective evolutionary algorithm: a preliminary study. *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 12-16 July 2008, Atlanta, 625-632.
- Duque, J. C., Church, R. L., Middleton, R. S., 2011. The p-regions problem. *Geographical Analysis*, 43 (1), 104-126.
- Emiliano, W. M., Telhada, J., and Carvalho, M. D., 2017. Home health care logistics planning: a review and framework. *Procedia Manufacturing*, 13, 948-955.
- Ferland, J. A., and Guenette, G., 1990. Decision Support System for the School Districting Problem. *Operations Research*, 38(1), 15-21.
- Forman, S. L. and Yue, Y., 2003. Congressional districting using a TSP-based genetic algorithm. *Lecture Notes on Computer Science*, 2724, 2072–2083.
- Franklin, A. D. and Koenigsberg, E., 1973. Computed School Assignments in a Large District. *Operations Research*, 21(2), 413-426.
- Garfinkel, R. S. and Nemhauser, G. L., 1970. Optimal Political Districting by Implicit Enumeration Techniques. *Management Science*, 16(8), 495-508.
- George, J. A., Lamar, B.W., and Wallace, C. A., 1997. Political district determination using large-scale network optimization. *Socioeconomic Planning Science*, 31(1), 11–28.
- Hess, S. W., *et al.*, 1965. Nonpartisan political redistricting by computer. *Operations Research*, 13 (6), 998–1006.
- Hojati, M., 1996. Optimal political districting. *Computers & Operations Research*, 23(12), 1147-1161.
- Horn, M. E. T., 1995. Solution Techniques for Large Regional Partitioning Problems. *Geographical Analysis*, 1995, 27: 230–248.
- Hu, F., Yang, S., and Xu, W., 2014. A non-dominated sorting genetic algorithm for the location and districting planning of earthquake shelters. *International Journal of Geographical Information Science*, 28(7), 1482-1501.
- Hu, Y., Wang, F., and Xierali, I., 2018. Automated Delineation of Hospital Service Areas and Hospital Referral Regions by Modularity Optimization. *Health Services Research*, 53(1), 236-255.
- Jacobs, D. A., Silan, M. N., and Clemson, B., 1996. An Analysis of Alternative Locations and Service Areas of American Red Cross Blood Facilities. *Interfaces*, 26(3), 40-50.
- Kalcsics, J., Nickel, S. and Schroder, M., 2005. Towards a unified territorial design approach: applications, algorithms and GIS integration. *Top*, 13(1), 1-56.
- Koenigsberg, E., 1968. Mathematical analysis applied to school attendance areas. *Socio-economic Planning Sciences*, 1(4): 465-475.
- Kong, Y., Zhu, Y., and Wang, Y., 2017. A hybrid metaheuristic algorithm for the school districting problem. *Acta Geographica Sinica*, 2017, 72(2): 256-268. (in Chinese)

- Kong, Y., Zhu, Y., and Wang, Y., 2019. A center-based modeling approach to solve the districting problem, *International Journal of Geographical Information Science*, 33(2), 368-384.
- Li, X., *et al.*, 2008. A Decentralized and Continuity-Based Algorithm for Delineating Capacitated Shelters' Service Areas. *Environment and Planning B: Planning and Design*, 35(4), 593–608.
- Li, Z., Wang, R. and Wang, Y., 2007. A quadratic programming model for political districting problem. In: X. Zhang, L. Chen, L. Wu, et al., eds. *The First International Symposium on Optimization and Systems Biology (OSB'07)*, 8–10 August 2007 Beijing. Beijing: World Publishing Corporation, 427-435.
- Liberatore, F., and Camachocollados, M., 2016. A Comparison of Local Search Methods for the Multicriteria Police Districting Problem on Graph. *Mathematical Problems in Engineering*, Article ID 3690474, 13 pages,
- Liu, Y., Cho, W. K., and Wang, S., 2016. PEAR: a massively parallel evolutionary computation approach for political redistricting optimization and analysis. *Swarm and evolutionary computation*, 30, 78-92.
- Lodi, A., 2017. *On Mixed-integer programming and its connection with data science* [online]. EPFL. Available from: <http://transp-or.epfl.ch/zinal/lectures2017.php> [Accessed 5 April 2018].
- Lourenço, H.R., Martin, O. and Stützle, T., 2010. Iterated Local Search: Framework and Applications. In: Gendreau, M. and Potvin, J.Y., eds. *Handbook of Metaheuristics*, 2nd. Edition. International Series in Operations Research & Management Science, Vol 146. New York: Springer, 363-397.
- Mehrotra, A., Johnson, E. L., and Nemhauser, G. L., 1998. An Optimization Based Heuristic for Political Districting. *Management Science*, 44(8), 1100-1114.
- Nemoto, T. and Hotta, K., 2003. Modelling and solution of the problem of optimal electoral districting. *Communications of Operations Research of Japan*, 48(4), 300-306 (in Japanese).
- Nygreen, B., 1988. European assembly constituencies for Wales—comparing of methods for solving a political districting problem. *Mathematical Programming*, 42(1), 159-169.
- Pezzella, F., Bonanno, R., and Nicoletti, B., 1981. A system approach to the optimal health-care districting. *European Journal of Operational Research*, 8(2), 139-146.
- Plane, D. A., Tong, D. and Lei T., 2019. Inter-person Separation: A New Objective Standard for Evaluating the Spatial Fairness of Political Redistricting Plans. *Geographical Analysis*, 51, 251–279
- Ricca, F. and Simeone, B., 2008. Local search algorithms for political districting. *European Journal of Operational Research*, 189(3), 1409-1426.

- Rincón-García E. A., *et al.*, 2015. ABC, A Viable Algorithm for the Political Districting Problem. In: Gil-Aluja J., *et al.* eds. *Scientific Methods for the Treatment of Uncertainty in Social Sciences. Advances in Intelligent Systems and Computing*, 377. Cham: Springer.
- Ríos-Mercado, R. Z., and Fernandez, E., 2009. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. *Computers & Operations Research*, 36(3), 755-776.
- Salazar-Aguilar, M. A., Ríos-Mercado, R. Z. and Gonzalez-Verlarde, J. L., 2011. A bi-objective programming model for designing compact and balanced territories in commercial districting. *Transportation Research Part C: Emerging Technologies*, 19(5), 885-895.
- Salazar-Aguilar, M. A., *et al.*, 2012. Multiobjective Scatter Search for a Commercial Territory Design Problem. *Annals of Operations Research*, 199(1), 343-360.
- Schoepfle, O. B. and Church, R. L., 1991. A new network representation of a "classic" school districting problem. *Socio-economic Planning Sciences*, 25(3), 189-197
- Shirabe, T., 2005. A model of contiguity for spatial unit allocation. *Geographical Analysis*, 37(1), 2-16.
- Sridharan, R., 1993. A Lagrangian heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research*, 66: 305-312
- Tavarespereira, F., *et al.*, 2007. Multiple criteria districting problems. *Annals of Operations Research*, 154(1), 69-92.
- Xiao, N., 2008. A Unified Conceptual Framework for Geographical Optimization Using Evolutionary Algorithms. *Annals of the Association of American Geographers*, 98(4), 795-817.
- Yanik, S., Surer, O., and Oztaysi, B., 2016. Designing sustainable energy regions using genetic algorithms and location-allocation approach. *Energy*, 97, 161-172.

Corresponding author: Yunfeng Kong, yfkong@henu.edu.cn

Foundation Support: The National Natural Science Foundation of China, No. 41871307.